AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions and listings of claims in the application.

Listing of Claims:

1. (previously presented) A computer-implemented method for extending functionality of a first set of classes and methods in an application hosted by a computing arrangement, comprising:

establishing a mapping of original class names of classes in the first set to corresponding substitute class names of classes in a second set, wherein the classes in the second set change the functionality of the classes in the first set;

establishing a mapping of original method names of the first set to corresponding substitute method names of methods in a second set, wherein the methods in the second set change the functionality of the methods in the first set;

in response to loading a class file of a class in the first set, replacing in the class file original methods names with corresponding substitute method names;

in response to loading a class file of a class in the first set, replacing in the class file original class names with corresponding substitute class names;

instantiating classes referenced by the substitute class names in lieu of classes referenced by the original class names;

caching in a class cache on the computing system the class file having the substitute class names and substitute method names; and

invoking methods referenced by the substitute method names in lieu of methods referenced by the original method names.

2. (canceled)

3. (canceled)

4. (currently amended) The method of claim 1 **[[3]]**, further comprising replacing the original class names with the substitute class names in a constant pool within the class

file, and changing in the class file method invocation bytecodes from references to original methods to references to substitute methods.

5. (currently amended) The method of claim 4, wherein the classes in the first set are selected ones of standard ~~Java~~ API classes of a platform independent programming language, and the classes in the second set change the functionality of the selected ones of the standard ~~Java~~ API classes.

6. (currently amended) The method of claim 5, wherein the selected ones of the standard ~~Java~~ API classes include selected input and output ~~Java~~ API classes.

7. (canceled)

8. (currently amended) An apparatus for extending functionality of a first set of classes and methods in an application hosted by a computing arrangement, comprising:

means for establishing a mapping of original class names of classes in the first set to corresponding substitute class names of classes in a second set, wherein the classes in the second set change the functionality of the classes in the first set;

means for establishing a mapping of original method names of the first set to corresponding substitute method names of methods in a second set, wherein the methods in the second set change the functionality of the methods in the first set;

means, responsive to loading a class file of a class in the first set, replacing in the class file original methods names with corresponding substitute method names;

means, responsive to loading a class file of a class in the first set, for replacing in the class file original class names with corresponding substitute class names; ~~and~~

means for instantiating classes referenced by the substitute class names in lieu of classes referenced by the original class names; ~~and~~

means for caching in a class cache on the computing system the class file having the substitute class names and substitute method names; and

means for invoking methods referenced by the substitute method names in lieu of methods referenced by the original method names.

9. (previously presented) A computing arrangement for extending functionality of a first set of classes and methods in an application program, comprising:

a second set of classes configured to change the functionality of the classes in the first set, wherein the second set of classes includes methods configured to change the functionality of the methods in the first set;

a mapping of original class names of classes in the first set to corresponding substitute class names of classes in the second set, wherein the mapping includes original method names of methods in the first set mapped to corresponding substitute method names of methods in the second set;

a virtual machine configured to host execution of the application program;

a class loader coupled to the virtual machine and responsive to a request from the virtual machine to load a class file of a class in the first set, the class loader configured to replace in class file original class names with corresponding substitute class names, whereby the virtual machine instantiates classes referenced by the substitute class names in lieu of classes referenced by the original class names, wherein the class loader is further configured to replace in class file original method names with corresponding substitute method names, whereby the virtual machine invokes methods referenced by the substitute method names in lieu of methods referenced by the original method names; and

a class cache coupled to the class loader and to the virtual machine for cache storage of the class file.

10. (canceled)

11. (canceled)

12. (currently amended) The arrangement of claim 9 **[[**11**]]**, wherein the class loader is further configured to replace the original class names with the substitute class names

in a constant pool within the class file, and change in the class file method invocation bytecodes from references to original methods to references to substitute methods.

13. (currently amended) The arrangement of claim 12, wherein the classes in the first set are selected ones of standard ~~Java~~ API classes of a platform independent programming language, and the classes in the second set change the functionality of the selected ones of the standard ~~Java~~ API classes.

14. (currently amended) The arrangement of claim 13, wherein the selected ones of the standard ~~Java~~ API classes include selected input and output ~~Java~~ API classes.

15. (canceled)

16. (currently amended) A method for processing a downloadable application program, comprising:

downloading the downloadable application program from a server system to a client system, wherein the downloadable application program includes a class file;

executing the downloadable application program on the client system;

loading a class file in response to execution of the downloadable application program;

determining from a first set of mapping data that includes ~~a set of~~ at least one association of an original class name to a substitute class name, each class name in the class file that is named as an original name in the mapping data;

replacing in the class file each reference to a class name named as an original class name with the associated substitute class name;

determining from a second set of mapping data that includes at least one association of an original method name to a substitute method name, each method name in the class file that is named as an original method name in the second set of mapping data, wherein a function implemented by a method with a substitute method name is different from a function implemented by a method with the associated original method name;

replacing in the class file each original method name having an associated substitute method name with the associated substitute method name;

caching the class file in a class cache on the client system after the replacing step; ~~and~~

during execution of the downloadable application program resolving each reference to a substitute class name in the class file; and

invoking methods referenced by the substitute method names in lieu of methods referenced by the original method names.

17. (canceled)

18. (previously presented) The method of claim 16, further comprising downloading a mapping file that contains the mapping data from the server system to the client system along with the downloading of the downloadable application program.

19. (previously presented) The method of claim 16, further comprising configuring the client system with the mapping data prior to the downloading of the downloadable application program.

20. (previously presented) The method of claim 16, further comprising statically configuring a class loader with the mapping data prior to downloading the downloadable application program.